종이접기

제37회 한국정보올림피아드 2차 대회(2020. 11. 15)

PLUG

문제

(중략) 한 변의 길이가 2^k인 정사각형 종이가 있을 때, 이를 세로로 k번, 가로로 k번 접으면 각 변의 길이가 1인 정사각형이 된다. 아래 그림에서 보인 것처럼 각 변의 길이가 1인 정사각형의 네 귀퉁이 중 한 군데에 구멍을 낸다.

구멍을 낸 후 접은 순서의 역순으로 종이를 펼치면, 2^{2k}개의 구멍이 있게 된다.

종이의 크기를 나타내는 정수 k, 종이를 접는 순서를 나타내는 정보, 구멍 뚫는 위치를 나타내는 정수가 주어질 때, 2^K X 2^K 격자에 뚫린 구멍의 위치를 출력하는 프로그램을 작성하시오. (중략)

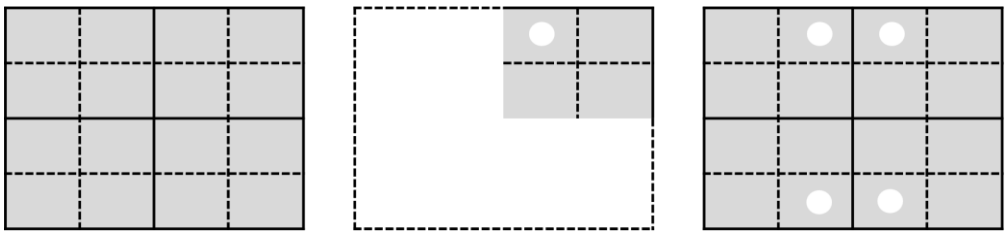
(종이를 접는 방법과 구멍 번호에 대한 설명은 이곳을 참고하세요.)

풀이

- 문제 설명

다음 왼쪽 그림과 같이 정사각형의 종이가 있다고 하자.

이를 오른쪽으로 한번, 위쪽으로 한번 접고 0번 구멍을 뚫으면 다음 중간 그림과 같이 된다. 그리고 이를 다시 펼치면 다음 오른쪽 그림이 되는데, 이러한 구멍들의 위치를 모두 출력하면 되는 것이다.

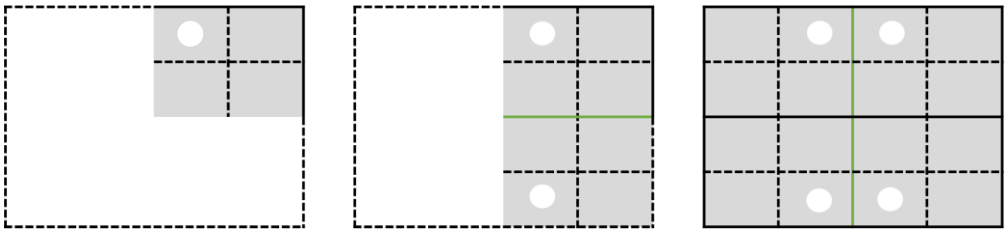


위 경우 다음을 출력하면 되는 것이다.

```
1 0
3 2
```

- 풀이 포인트

위 예제의 종이 펼치는 상황을 더 자세히 살펴보자,



종이를 펼칠 때 마다 어떠한 선을 기준으로 구멍이 대칭을 이루는 것을 알 수 있다.

- 문제 풀이

위 대칭을 이룬다는 사실을 이용해 구멍의 위치를 기록하고, 기록을 통해 어느 숫자에 구멍이 뚫렸는지 출력하면 되는 것이다.

코드

- 풀이 환경

Visual Studio 2022, C

- 코드

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main() {

	int K, h; // K와 처음 구멍을 뚫는 위치 h 정의.

	int location[512][512] = { 0 }; // 구멍 뚫린 위치를 기록할 2차원 배열.
	int final_hole[256][256] = { 0 }; // 최종적으로 출력할 구멍 위치 2차원 배열.
	char fold[17] = { 0 }; // 접는 순서를 기록할 배열.

	scanf("%d", &K); // K 입력받음.

	// 제곱으로 종이 길이 구하기
	int paper_length = 1;

	for (int x = 0; x < K; x++) {
		paper_length = paper_length * 2;
	}

	int x[2] = { 0, (paper_length * 2 - 1) }, y[2] = { 0, (paper_length * 2 -
	1) }; // 현재 접힌 부분이 차지하는 위치를 나타낼 x, y 좌표 정의.
```

```
	for (int i = 0; i < (K * 2); i++) {
		scanf("%s", &fold[i]); // 접는 방법 입력받음.
	}

	scanf("%d", &h); // h 입력받음.

	// 접힌 위치 기록
	for (int j = 0; j < (K * 2); j++) {
		if (fold[j] == 'U') {
			y[1] = y[1] - ((y[1] - y[0] + 1) / 2);
		}
		if (fold[j] == 'D') {
			y[0] = y[0] + ((y[1] - y[0] + 1) / 2);
		}
		if (fold[j] == 'L') {
			x[1] = x[1] - ((x[1] - x[0] + 1) / 2);
		}
		if (fold[j] == 'R') {
			x[0] = x[0] + ((x[1] - x[0] + 1) / 2);
		}
	}

	// 처음 구멍 뚫는 위치 기록
	if (h == 0) {
		location[y[0]][x[0]] = 1;
	}
	if (h == 1) {
		location[y[0]][x[1]] = 1;
	}
	if (h == 2) {
		location[y[1]][x[0]] = 1;
	}
	if (h == 3) {
		location[y[1]][x[1]] = 1;
	}

	// 종이 펼치기.
	for (int s = 0; s < (K * 2); s++) {
		if (fold[(K * 2) - 1 - s] == 'U') {

			// 코드 해설 2 - 1 참고.
			for (int o = x[0]; o < x[1] + 1; o++) {
				for (int a = y[0]; a < y[1] + 1; a++) {
					location[y[1] + (y[1] - y[0] + 1) - (a -
					y[0])][o] = location[a][o];
				}
			}

			y[1] = y[1] + (y[1] - y[0] + 1);

		}
		if (fold[(K * 2) - 1 - s] == 'D') {

			// 코드 해설 2 - 2 참고.
			for (int o = x[0]; o < x[1] + 1; o++) {
				for (int a = y[0]; a < y[1] + 1; a++) {
					location[y[0] - (a - y[0] + 1)][o] =
					location[a][o];
				}
			}

			y[0] = y[0] - (y[1] - y[0] + 1);
		}
```

```
                if (fold[(K * 2) - 1 - s] == 'L') {

                        // 코드 해설 2 - 3 참고.
                        for (int o = y[0]; o < y[1] + 1; o++) {
                                for (int a = x[0]; a < x[1] + 1; a++) {
                                        location[o][(x[1] + (x[1] - x[0] + 1)) -
                                        (a - x[0])] = location[o][a];
                                }
                        }

                        x[1] = x[1] + (x[1] - x[0] + 1);

                }
                if (fold[(K * 2) - 1 - s] == 'R') {

                        // 코드 해설 2 - 4 참고.
                        for (int o = y[0]; o < y[1] + 1; o++) {
                                for (int a = x[0]; a < x[1] + 1; a++) {
                                        location[o][(x[0] - (a - x[0] + 1))] =
                                        location[o][a];
                                }
                        }

                        x[0] = x[0] - (x[1] - x[0] + 1);

                }
        }

        // 구멍 번호 기록하기
        for (int q = 0; q < (paper_length * 2); q++) {
                for (int p = 0; p < (paper_length * 2); p++) {
                        if (q == 0 || q % 2 == 0) {
                                if (p == 0 || p % 2 == 0) {
                                        if(location[q][p] == 1) final_hole[q /
                                                2][p / 2] = 0;
                                }
                                else {
                                        if(location[q][p] == 1) final_hole[q /
                                                2][p / 2] = 1;
                                }
                        }
                        else {
                                if (p == 0 || p % 2 == 0) {
                                        if(location[q][p] == 1) final_hole[q /
                                                2][p / 2] = 2;
                                }
                                else {
                                        if(location[q][p] == 1) final_hole[q /
                                                2][p / 2] = 3;
                                }
                        }
                }
                if (q % 2 != 0) {
                        // 구멍 번호 출력하기
                        for (int p = 0; p < (paper_length); p++) {
                                printf("%d ", final_hole[q / 2][p]);
                        }
                        if(q != (paper_length * 2 - 1)) printf("\n");
                }
        }

        return 0;

}
```

- 코드 풀이

이번 코드에는 수식이 꽤 들어가는 편이다.

2차원 배열에 관해 충분한 이해를 가지고 있다고 가정하고 설명한다.

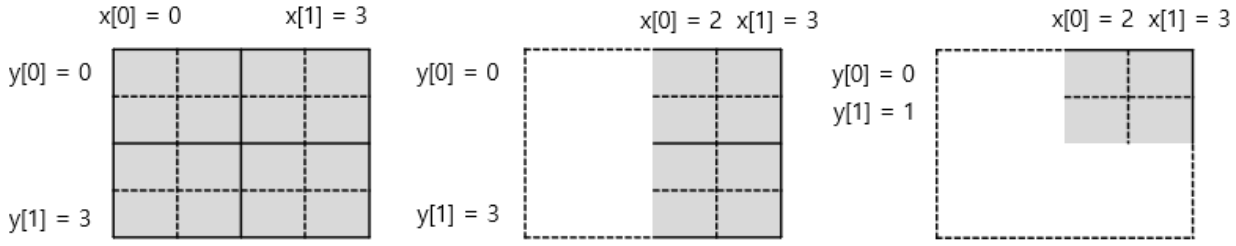
구멍의 위치를 기록하기 위해 2차원 배열을 사용했다.

-> 1. 접힌 위치 기록하기

접힌 위치 기록하기 코드를 보면, 접힌 부분의 위치 구간을 X와 Y좌표로 x, y 변수에 기록하고 있다. 처음에 x, y 변수의 값을 0 ~ (paper_length * 2 - 1)으로 초기화했는데,

이는 0 ~ 2^k(처음 종이의 길이) X 2(최종적으로 접은 종이를 4칸으로 나눠 한 칸을 뚫기 때문) – 1(배열은 0부터 시작하기 때문)와 같다.

위 예제에서 접힌 위치를 기록하면서 x, y 변수의 값은 다음과 같은 과정으로 변하게 된다.



-> 2. 처음 구멍을 뚫는 위치 기록

접힌 위치를 기록하고 나면 처음 구멍의 위치(0 ~ 3)과 x, y 변수를 이용해 2차원 배열에 어느 구멍이 뚫렸는지 1로 기록했다. 지금까지 구멍 위치 2차원 배열의 값은 다음과 같다.

0	0	1	0
0	0	0	0
0	0	0	0
0	0	0	0

-> 3. 종이 펼치기

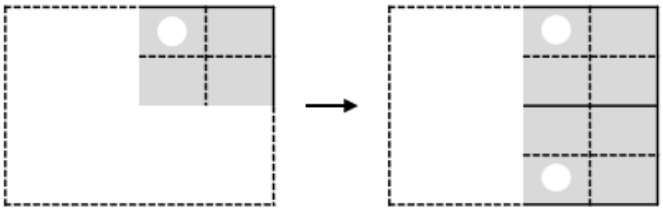
가장 중요한 부분이다.

펼칠 때마다 일정한 선을 기준으로 구멍이 대칭을 이뤄야 하는데,

펼칠 때마다 반복문을 이용해서 일정한 선을 기준으로 펼치기 전에 있던 부분을 펼쳐지는 부분에 복사한다고 생각한다.

또한, 펼치고 나서는 펼쳐진 부분이 어디인지 x, y 변수를 변경한다.

펼치는 과정 중 첫번째에서 같은 색이 같은 값을 같도록 복사한다고 생각하면, 다음과 같다.



0	0	1	0
0	0	0	0
0	0	0	0
0	0	0	0

0	0	1	0
0	0	0	0
0	0	0	0
0	0	1	0

0	0	1	0
0	0	0	0
0	0	0	0
0	0	1	0

0	0	1	0
0	0	0	0
0	0	0	0
0	0	1	0

위와 같이 아래로 다시 펼치는 상황에서는 x값을 한번 고정시키고 y값을 올리면서 복사하고, 다시 x값을 바꿔 반복하며 복사하는 식이다.

-> 4. 구멍 기록하기

구멍 기록하기 단계에서는 반복문으로 다음과 같은 배열의 값을 모두 둘러보며 x, y 좌표가 홀수인지 짝수 인지 0인지에 따라 구멍의 번호가 몇 번인지 인지하고, 해당 구멍의 번호를 기록한다.

이때 반복하고 있는 y 값이 짝수이면 한 줄을 모두 기록했다는 의미가 되므로,

해당 줄의 구멍 번호를 출력한다.

0	1	1	0
0	0	0	0
0	0	0	0
0	1	1	0